Discovery of Nominally Conditioned Polynomials using Neural Networks, Vector Quantizers and Decision Trees

Kazumi Saito[1] and Ryohei Nakano[2]

[1] NTT Communication Science Laboratories
2-4 Hikaridai, Seika, Soraku, Kyoto 619-0237 Japan
saito@cslab.kecl.ntt.co.jp
[2] Nagoya Institute of Technology
Gokiso-cho, Showa-ku, Nagoya 466-8555 Japan
nakano@ics.nitech.ac.jp

1 Introduction

The discovery of a numeric law, e.g., Kepler's third law $T = kr^{3/2}$, from data is the central part of scientific discovery systems. The data considered in many real fields usually contains both nominal and numeric values. Thus, we consider discovering a law that governs such data in the form of a rule set of nominally conditioned polynomials. Recently, a connectionist method called RF6 [1] was proposed to solve problems of this type. RF6 can learn multiple nominally conditioned polynomials with single neural networks; besides, RF6 can discover generalized polynomials whose power values are not restricted to integers. However, for real complex problems, RF6 will suffer from a combinatorial explosion in the process of restoring rules from a trained neural network. Therefore, this paper proposes a new version of RF6 by greatly improving its procedure of restoring nominally conditioned polynomials from a trained neural network.

2 Restoring Nominally Conditioned Polynomials

Let $\{q_1, \cdots, q_{K_1}, x_1, \cdots, x_{K_2}, y\}$ be a set of variables describing an example, where q_k is a nominal explanatory variable, x_k is a numeric explanatory variable and y is a numeric target variable. For each q_k we introduce a *dummy variable* expressed by q_{kl}, i.e., $q_{kl} = 1$ if q_k matches the l-th category; otherwise 0, where $l = 1, \cdots, L_k$, and L_k is the number of distinct categories appearing in q_k. As a true model governing data, we consider the following set of multiple *nominally conditioned polynomials* whose power values are not restricted to integers.

$$if \bigwedge_{q_{kl} \in Q^i} q_{kl} = 1 \quad then \;\; y(\boldsymbol{x}; \boldsymbol{\Theta}^i) = w_0^i + \sum_{j=1}^{J^i} w_j^i \prod_{k=1}^{K_2} x_k^{w_{jk}^i}, \quad i = 1, \cdots, I^* \quad (1)$$

where I^* is the number of rules, Q^i denotes a set of dummy variables corresponding to the i-th nominal condition and $\boldsymbol{\Theta}^i$ is a parameter vector used in the i-th generalized polynomial. Here, each parameter w_j^i or w_{jk}^i is a real number, and J^i is an integer corresponding to the number of terms.

Consider a function $c(\boldsymbol{q};\boldsymbol{V}) = \exp(\sum_{k=1}^{K_1}\sum_{l=1}^{L_k} v_{kl}q_{kl})$, where $\boldsymbol{V}$ denotes a vector of parameters v_{kl}. We can show [1] that with an adequate number J, certain type of neural network $y(\boldsymbol{q},\boldsymbol{x};\boldsymbol{\Theta}) = w_0 + \sum_{j=1}^{J} w_j \exp(\sum_{k=1}^{K_1}\sum_{l=1}^{L_k} v_{jkl}\, q_{kl} + \sum_{k=1}^{K_2} w_{jk} \ln x_k)$ can closely approximate Eq. (1). Let $D = \{(\boldsymbol{q}^\mu, \boldsymbol{x}^\mu, y^\mu) : \mu = 1,\cdots,N\}$ be a set of training data, where N is the number of examples. Then, each parameter can be estimated by minimizing an objective function $\mathcal{E}(\boldsymbol{\Theta}) = \sum_{\mu=1}^{N}(y^\mu - y(\boldsymbol{q}^\mu,\boldsymbol{x}^\mu;\boldsymbol{\Theta}))^2 + \boldsymbol{\Theta}^T\boldsymbol{\Lambda}\boldsymbol{\Theta}$, where a penalty term is added to improve both the generalization performance and the readability of the learning results.

2.1 Restoring Procedures

Assume that we have already obtained a neural network trained as the best law-candidate. In order to restore a set of nominally conditioned polynomials as described in Eq. (1), we need a suitable efficient procedure.

RF6 [1] has a decomposition procedure for this purpose; i.e., a set of nominally conditioned terms is extracted from each hidden unit, and then each of these terms is in turn combined through all of the hidden units. When α denotes the average number of terms over each hidden unit, the total number of these combined terms approximately amounts to α^J. Thus, as the number of hidden units or the number of nominal variables increases, this procedure comes to suffer from a combinatorial explosion.

As another approach, we can extract nominally conditioned polynomials for each training example, and simply assemble them to obtain a final set of rules as a law. Then, the following set of nominally conditioned polynomials can be obtained directly from the training data and the trained neural network.

$$if \bigwedge_{\{k,l:q_{kl}^\mu=1\}} q_{kl} = 1 \quad then \;\; y = \widehat{w}_0 + \sum_{j=1}^{J} c_j^\mu \prod_{k=1}^{K2} x_k^{\widehat{w}_{jk}}, \quad \mu = 1,\cdots,N, \qquad (2)$$

where c_j^μ denotes the j-th coefficient calculated from the nominal values of the μ-th training example, i.e., $c_j^\mu = \widehat{w}_j \exp(\sum_{k=1}^{K_1}\sum_{l=1}^{L_k} \widehat{v}_{jkl} q_{kl}^\mu)$. However, in comparison with the true model governing the data defined in Eq. (1), the results of this naive procedure can be far from desirable because they will contain a large number of similar polynomials, and each nominal condition will be too specific in terms of representing only one training example.

Based on the above considerations, we propose a new restoring procedure.

step1. finding subspace representatives

In order to find subspace representatives, a set of coefficient vectors $\{\boldsymbol{c}^\mu = (c_1^\mu,\cdots,c_J^\mu)^T : \mu = 1,\cdots,N\}$ calculated from the training data is quantized into a set of representative vectors $\{\boldsymbol{r}^i = (r_1^i,\cdots,r_J^i)^T : i = 1,\cdots,I\}$, where I is the number of representatives. Among several vector quantization (VQ) algorithms, we employ the k-means algorithm due to its simplicity.

step2. criterion for model selection

Consider the following set of rules using the representative vectors.

$$if\;\, i(\boldsymbol{q}) = i \quad then \;\; y = \widehat{w}_0 + \sum_{j=1}^{J} r_j^i \prod_{k=1}^{K2} x_k^{\widehat{w}_{jk}}, \quad i = 1,\cdots,I, \qquad (3)$$

where $i(\boldsymbol{q})$ denotes a function that returns the index of the representative vector minimizing the distance, i.e., $i(\boldsymbol{q}) = \arg\min_i \sum_{j=1}^{J} (c_j - r_j^i)^2$. Here, since each element of $\boldsymbol{c}$ is calculated as $c_j = \widehat{w}_j \exp(\sum_{k=1}^{K_1} \sum_{l=1}^{L_k} \widehat{v}_{jkl} q_{kl})$, Eq. (3) can be applied to a new example, as well as the training examples. Thus, to determine an adequate number I of representatives, we employ the procedure of cross-validation which divides the data D at random into S distinct segments $\{D^s : \ s = 1, \cdots, S\}$. Namely, by using the final weights $\widehat{\boldsymbol{\Theta}}^{(s)}$ trained without data segment D^s, we can define a cross-validation error function $\mathcal{CV} = N^{-1} \sum_{s=1}^{S} \sum_{\nu \in D_s} (y^\nu - (\widehat{w}_0^{(s)} + \sum_{j=1}^{J} r_j^{i(\boldsymbol{q}^\nu)} \prod_{k=1}^{K2} (x_k^\nu)^{\widehat{w}_{jk}^{(s)}}))^2$.
step 3. generating conditional parts
The indexing functions $\{i(\boldsymbol{q})\}$ described in Eq. (3) must be transformed into a set of nominal conditions as described in Eq. (1). One reasonable approach is to perform this transformation by solving a classification problem whose training examples are $\{(\boldsymbol{q}^\mu, i(\boldsymbol{q}^\mu)) : \mu = 1, \cdots, N\}$, where $i(\boldsymbol{q}^\mu)$ indicates the class label of a training example $\boldsymbol{q}^\mu$. For this classification problem, we employ the c4.5 decision tree generation program due to its wide availability. From the generated decision tree, we can easily obtain the final rule set as described in Eq. (1).

Clearly, these steps can be executed within the computational complexity of linear order with respect to the numbers of training examples, variables, hidden units, representatives, iterations performed by the k-means algorithm, and data segments used concerning cross-validation; i.e., this new restoring procedure can be much more efficient than the old decomposition procedure which requires the computational complexity of exponential order. Hereafter, the law discovery method using the above restoring procedure is called *RF6.2*.

2.2 Evaluation by Experiments

By using three data sets, we evaluated the performance of RF6.2. In the k-means algorithm, initial representative vectors $\{\boldsymbol{r}^i\}$ are randomly selected as a subset of coefficient vectors $\{\boldsymbol{c}^\mu\}$. For each I, trials are repeated 100 times with different initial values, and the best result is reported. The cross-validation error is calculated by using the leave-one-out method, i.e., $S = N$. The candidate number I of representative vectors is incremented in turn from 1 until the cross-validation error increases. The c4.5 program is used with the initial settings.
Artificial data set.
We consider an artificial law described by

$$\begin{cases} if \quad q_{21} = 1 \wedge (q_{31} = 1 \vee q_{33} = 1) & then \quad y = 2 + 3x_1^{-1}x_2^3 + 4x_3x_4^{1/2}x_5^{-1/3} \\ if \quad q_{21} = 0 \wedge (q_{32} = 1 \vee q_{34} = 1) & then \quad y = 2 + 5x_1^{-1}x_2^3 + 2x_3x_4^{1/2}x_5^{-1/3} \\ else & then \quad y = 2 + 4x_1^{-1}x_2^3 + 3x_3x_4^{1/2}x_5^{-1/3} \end{cases} \quad (4)$$

where we have three nominal and nine numeric explanatory variables, and the numbers of categories of q_1, q_2 and q_3 are set as $L_1 = 2$, $L_2 = 3$ and $L_3 = 4$, respectively. Clearly, variables $q_1, x_6, \cdots, x_9$ are irrelevant to Eq. (4). Each value of nominal variables q_1, q_2, q_3 is randomly generated so that only one dummy variable becomes 1, each value of numeric variables $x_1, \cdots, x_9$ is randomly generated in the range of $(0, 1)$, and we get the corresponding value of y by calculating

Eq. (4) and adding Gaussian noise with a mean of 0 and a standard deviation of 0.1. The number of examples is set to 400.

In this experiment, a neural network was trained by setting the number of hidden units J to 2. We examined the performance of the experimental results obtained by applying the k-means algorithm with the different number of representative vectors, where the RMSE (root mean squared error) was used for the evaluation; the training error was evaluated as a rule set by using Eq. (3); the cross-validation error was calculated by using the function $\mathcal{CV}$; and the generalization error was also evaluated as a rule set and measured by using a set of noise-free $10,000$ test examples generated independently of the training examples. The experimental results showed that the training error almost monotonically decreased (2.090, 0.828, 0.142, and 0.142 for I = 1, 2, 3, and 4, respectively); the cross-validation error was minimized when $I = 3$ (2.097, 0.841, 0.156, and 0.160 for I = 1, 2, 3, and 4, respectively, i.e., indicating that an adequate number of representative vectors is 3); and the generalization error was also minimized when $I = 3$ (2.814, 1.437, 0.320, and 0.322 for I = 1, 2, 3, and 4, respectively). Since the cross-validation and generalization errors were minimized with the same number of representative vectors, we can consequently see that the desirable model was selected by using the cross-validation.

By applying the c4.5 program, we obtained the following decision tree whose leaf nodes correspond to the following.

$$
\begin{array}{llll}
q_{21} = 0\text{:} & q_{34} = 1\text{: } 2\ (83.0) & \Leftrightarrow & \boldsymbol{r}^2 = (+5.04, +2.13) \\
| & q_{34} = 0\text{:}\ \ q_{32} = 0\text{: } 3\ (129.0) & \Leftrightarrow & \boldsymbol{r}^3 = (+3.96, +2.97) \\
| & |\qquad\quad\ q_{32} = 1\text{: } 2\ (53.0) & \Leftrightarrow & \boldsymbol{r}^2 = (+5.04, +2.13) \\
q_{21} = 1\text{:} & q_{34} = 1\text{: } 3\ (36.0) & \Leftrightarrow & \boldsymbol{r}^3 = (+3.96, +2.97) \\
| & q_{34} = 0\text{:}\ \ q_{32} = 0\text{: } 1\ (73.0) & \Leftrightarrow & \boldsymbol{r}^1 = (+3.10, +4.07) \\
| & |\qquad\quad\ q_{32} = 1\text{: } 3\ (26.0) & \Leftrightarrow & \boldsymbol{r}^3 = (+3.96, +2.97)
\end{array}
$$

where the coefficient values were rounded off to the second decimal place; each number of training examples arriving at the corresponding leaf node is shown in parenthesis. Then, the following rule set was straightforwardly obtained.

$$
\begin{cases}
\ \ \textit{if}\ \ q_{21} = 1 \wedge (q_{31} = 1 \vee q_{33} = 1) \\
\qquad \textit{then}\ y = 2.01 + 3.10 x_1^{-1.00} x_2^{+3.01} + 4.07 x_3^{+1.02} x_4^{+0.51} x_5^{-0.33} \\
\ \ \textit{if}\ \ q_{21} = 0 \wedge (q_{32} = 1 \vee q_{34} = 1) \\
\qquad \textit{then}\ y = 2.01 + 5.04 x_1^{-1.00} x_2^{+3.01} + 2.13 x_3^{+1.02} x_4^{+0.51} x_5^{-0.33} \\
\textit{else}\ \ \textit{then}\ y = 2.01 + 3.96 x_1^{-1.00} x_2^{+3.01} + 2.97 x_3^{+1.02} x_4^{+0.51} x_5^{-0.33}.
\end{cases} \tag{5}
$$

Recall that each nominal variable matches only one category, e.g., $(q_{32} = 1 \wedge q_{34} = 0) \equiv (q_{32} = 1)$. Therefore, although some of the weight values were slightly different, we can see that a law almost equivalent to the true one was found.

Financial data set.

We performed an experimental study to discover underlying laws of market capitalization from six fundamental BS (Balance Sheet) items and the type of industry (the 33 classifications of the Tokyo Stock Exchange). Our experiments used data from 953 companies listed on the first section of the TSE, where banks, and insurance, securities and recently listed companies were excluded. In order to understand the effect of the nominal variable intuitively, the number of hidden units was fixed at 1. The cross-validation error was minimized at $I = 3$. Then, the following rule set was obtained.

$$if \bigvee_{q_l \in Q^i} q_l = 1 \; then \;\; y = 12891.6 + r^i x_2^{+0.668} x_3^{+1.043} x_6^{-0.747}, \quad i = 1, 2, 3 \tag{6}$$

where $r^1 = +1.907$, $r^2 = +1.122$, $r^3 = +0.657$ and each of the nominal conditions was as follow: $Q^1 = \{$"Pharmaceuticals", "Rubber Products", "Metal Products", "Machinery", "Electrical Machinery", "Transport Equipment", "Precision Instruments", "Other Products", "Communications", "Services"$\}$; $Q^2 = \{$"Foods", "Textiles", "Pulp & Paper", "Chemicals", "Glass & Ceramics", "Nonferrous Metals", "Maritime Transport", "Retail Trade"$\}$; and $Q^3 = \{$"Fisheries", "Mining", "Construction", "Oil & Coal Products", "Iron & Steal", "Electricity & Gas", "Land Transport", "Air Transport", "Wearhousing", "Wholesale", "Other Financing Business", "Real Estate"$\}$. Since the second term on the right hand side of the polynomials appearing in Eq. (6) is always positive, each of the coefficient values r^i can indicate the stock price setting tendency of industry groups in similar BS situations, i.e., the discovered law tells us that industries appearing in Q^1 are likely to have a high setting, while those in Q^3 are likely to have a low setting.

Automobile data set.
The Automobile data set contained data on the car and truck specifications in 1985, and was used to predict prices based on these specifications. The data set had 159 examples with no missing values, and consisted of 10 nominal and 14 numeric explanatory variables and one target variable (price). In this experiment, since the number of examples was small, the number of hidden units was also set to 1. The cross-validation error was minimized at $I = 3$. The polynomial part of the discovered law was as follows:

$$y = 1163.16 + r^i x_2^{+1.638} x_4^{+0.046} x_5^{-1.436} x_6^{+0.997} x_9^{-0.245} x_{13}^{-0.071} \tag{7}$$

where $r^1 = +1.453$, $r^2 = +1.038$, $r^3 = +0.763$ and the relatively simple nominal conditions were obtained. Similarly as described for the experiments using the financial data set, since the second term on the right hand side of Eq. (7) is always positive, the coefficient value r^i can indicate the car price setting tendency for similar specifications. Actually, the discovered law verbally told us that cars of a high price setting are: "5-cylinder ones", "BMW's", "convertibles", "VOLVO turbos", "SAAB turbos", and "6-cylinder turbos"; cars of a middle price setting are: "PEUGOT's", "VOLVO non-turbos", "SAAB non-turbos", "HONDA 1bbl-fuel-system ones", "MAZDA fair-risk-level ones", "non-BMW non-turbos & 6-cylinder ones", "non-5-cylinder turbos & fair-risk-level ones"; and other cars are of a low price setting.

Reference

1. R. Nakano and K. Saito. Discovery of a set of nominally conditioned polynomials. In *Proc. 2nd Int. Conference on Discovery Science, LNAI 1721*, pp. 287–298, 1999.